BGP-ASPA Hackathon Report

IETF 112

Hackathon Team:
O. Borchert, L. Hannachi, D. Montgomery, K. Sriram
Goal

• Develop tools and data sets to facilitate testing emerging BGP route-leak mitigation technique
What tools did we use?

• We use the NIST BGP-SRx Software Suite V6 that provides reference implementation for:

  • draft-ietf-sidrops-aspa-verification-08+ (update with algorithm correction*)
  • draft-ietf-sidrops-8210bis-03
  • and test harnesses that enables scripted experiments with RPKI & BGP data sets.

• Source: https://github.com/usnistgov/NIST-BGP-SRx

High Level ASPA Data Flow
Develop tools and data sets for testing router implementations of ASPA. Unit tests and Internet scale tests

Task 1:
- Create sample Internet scale ASPA data set for use with 8210bis-03 using CAIDA reference data
- Use SRx Test harness ASCII format:
  addASPA <AFI> <CustomerAS> <ProviderAS>+

Task 2:
- Create sample BGP UPDATES using data from RouteViews3.
What got done

• We designed a test framework that allows to generate
  • CAIDA based ASPA script data describing 72,616 ASPA PDU’s containing 148,284 customer provider relations.
  • Data Pool is down-selectable to only use ASPA link relations for ASN’s found within UPDATE stream only
  • Specified a result output that can be used to compare between implementations
• Created Data Sets 100, 500, 800, 1K, 10K, and 20K unique AS PATHs using RouteViews and CAIDA Data
1. Preparation of BGP peers from RouteViews3 Data Set for BGPsec-IO

- We generated UPDATE traffic files, one for each peer containing the UPDATE send to the collector
- We removed the Peer AS (will be added by the player again)
- We added the marker B4 BGPsec-IO to only generate BGP-4 UPDATES and NOT BGPsec UPDATES
Convert all CAIDA Data to BGP-SRx Cache Test Harness Format

- To use the “rpkirtr_svr” BGP-SRx cache test harness we needed the CAISA data in the following script style: addASPA <afi> <customer> <provider>+

- We generated a total of 72,616 ASPA data entries with 148,284 link relations.
We specify the peer and the maximum UPDATES

• Here we down select the peers UPDATES to “X” UNIQUE AS Paths and removed the prefix.

• We added a synthetic generated prefix from the prefix pool 0.0.1.0/24 to 255.255.255.0/24 to assure no path uses the same prefix.*

* Can happen if raw data comes from UPDATE stream and not RIB in.
The ASPA data is generated depending on the UPDATE traffic.

- From the selected UPDATE traffic a list of all unique ASes is generated
- From the 72K available Customer specification only those ASes are selected that found in the UPDATE traffic.
- A downsized ASPA data file is generated

Creation of APSA Test Data
Starting the Experiment

- Once the experimental data is generated, the starter script allows two modes:
  - Terminal Only
    - In this mode each module is started in the background
    - All output standard and error is redirected into log files.
  - Gnome Terminal
    - This mode is preferable for window based Linux systems
    - Here each module will be started in its own terminal tab
      - In case something goes wrong, this mode is simpler to debug.
      - This mode allows to control the cache test harness
The Gnome Terminal Mode

Each Module is started in its own terminal window

Result output
The Experiment

We used RouteViews-3 BGP data, Large Scale ISP and CAIDA data from Oct. 1, 2020

• We created a subset of unique routes.

• We selected only CAIDA data where ASN in each path is listed as customer

• Then we performed ASPA validation

• IUT is private ASN peering with Large Scale ISP
Some Results

<table>
<thead>
<tr>
<th>ISP is Provider of IUT</th>
<th>Valid</th>
<th>invalid</th>
<th>unknown</th>
<th>unverifiable</th>
</tr>
</thead>
<tbody>
<tr>
<td>94%</td>
<td>3%</td>
<td>3%</td>
<td>0%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ISP is Customer of IUT</th>
<th>Valid</th>
<th>invalid</th>
<th>unknown</th>
<th>unverifiable</th>
</tr>
</thead>
<tbody>
<tr>
<td>14%</td>
<td>18%</td>
<td>68%</td>
<td>0%</td>
<td></td>
</tr>
</tbody>
</table>

Results vary from peer to peer
The Code

- We still refine the code and then will publish it once it's ready
  - Once published we will provide the location of the framework on the list
  - Also we will provide a link in our GitHub page for NIST BGP-SRx V6:
Future Work / Hackathons

• More experiments to study gradual deployment of ASPA objects
  • Selecting different peers
• For proper performance testing extending framework to use multiple peering sessions
  • Manual possible but it would be nice to have it automated as well
  • Scaling, scaling, scaling,....
• Other implementations to test against
  • Maybe next hackathon
• Create ASPA objects for testing Validation Caches?
  • Maybe others can join in!
Questions?

Oliver Borchert
oliver.borchert@nist.gov

General Questions:
itrzg-contact@list.nist.gov